A GCN-based Neural Network For Toxicity Prediction On SMILES

Shengyuan Hou
Shanghai JiaoTong University
Shanghai JiaoTong University, No.800 Dongchuan Road, Minhang District, Shanghai, China
445164577@qq.com

Abstract

Nowadays explosive information has been mined about chemical properties and structures, and to better store and retrieve them, chemical molecule structures are often transformed into representations of text-like string, SMILES, and their properties are manually explored and stored in fingerprints. For example, pharmaceutical enterprise needs to timely and effectively ensure the toxicity of one drug. However, as more and more new molecule compounds are synthetized, it's often time and cost consuming for artificial feature engineering, which could not adapt to the rapid changes of the market. With the development of deep learning techniques, it provides a new handy method to predict molecular toxicity. In this work, we develop a popular GCN nerual network structure with an addition of embedding layer, absorbing the atomic properties and categories as the input. Our auc result overcomes 0.9 and excels most of the competitors on kaggle competitoin.

1 Introduction

Nowadays, as our society comes into the big data era, explosive data has been mined, stored, retrieved and represented. In chemistry field, traditionally limited by relatively lagging scientific techniques, recognitions of chemical compounds such as their structures and properties are finite and are explored by artificial experiments in the past. Nevertheless, along with the progress on the computational capability and big data mining recently, chemists are endowed to predict molecuar features by studying the relationship between molecules' structures and properties from methematical models. Since compounds are often represented with graph structures in traditoinal chemical study , including the atomic type and different kinds of chemical bonds, which are not fit in big database storage and as the input of deep learning model, a new text-like string representation of molecules, SMILES, has been introduced and it's handy to store, transfer and quantinization. Ihis enables us to regard it as a text sentence and introduce NLP(Natual Language Processing) techniques to recognize its properties, such as the embeding layer technique.

1.1 SMILES representation

SMILES (Simplified molecular input line entry specification), a simplified molecular linear input specification, is a specification that uses ASCII strings to clearly describe the molecular structure. SMILES was developed by Arthur Weininger and David Weininger in the late 1980s, and was modified and extended by others, especially Daylight Chemical Information Systems Inc. (Daylight Chemical Information Systems Inc.).

Since SMILES uses a string of characters to describe a three-dimensional chemical structure, it must convert the chemical structure into a spanning tree. This system uses a vertical first traversal tree algorithm. During conversion, hydrogen must be removed first, and the ring must be opened. When

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

indicating, the atom at the end of the bond to be removed must be marked with a number, and the branch is written in parentheses.

SMILES strings can be imported by most molecular editing software and converted into 2D graphics or 3D molecular models.

```
CC=CCC#N
OC1=C(Br)C=C(Br)C=C1C(=O)NC2=CC=C(Br)C=C2
[O-][N+](=O)C1=CC(=C(Cl)C=C1)[N+]([O-])=O
[Na+].CN(C)C(=O)CCSC(SCCC([O-])=O)C1=CC(\C=C\C2=NC3=CC(Cl)=CC=C3C=C2)=CC=C1
Cl.CCOC(=O)COC1=CC2=C(CCC(C2)NCC(O)C3=CC(Cl)=CC=C3)C=C1
CN(C)C(Cl)=O
COC(=O)CS
CCC\C=C\CO
CCCCSP(=O)(SCCCC)SCCCC
Cl.OCCOCCN1CCN(CC1)C(C2=CC=CC=C2)C3=CC=C(Cl)C=C3
O.O.[Na+].[O-]C1=NC(=O)N(Cl)C(=O)N1Cl
O=C(C1CCCCC1)N2CC3N(CCC4=C3C=CC=C4)C(=O)C2
COC(C)(C)C
C(OC1=C(C=CC=C1)C2CC2)C3=NCCN3
```

Figure 1: SMILES Representation

1.2 GCN

GCN(Graph-based convolutional neural network) is a newly invented convolutional neural network structure[5,6], which could contain the spatial structure of the input and this is really important for the chemial property prediction of molecules. Traditional CNN is often equipped with an input layer, hidden layer and output layer. The hidden layer is composed of a convolutional layer with kernel, pooling layer and fully connected layer. They are broadly adapted in the computer vision, natural language processing and so on. However, their performance are not so satisfactory in the graph-structured data. GCN takes advantage of the adjacent matrix and degree matrix of the graph structure to calculate the matrix $\hat{A}$ which contains the spatial information.

$$\hat{A} = D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}$$

$$Z = f(X, A) = Softmax(ReLU(\hat{A}XW))$$

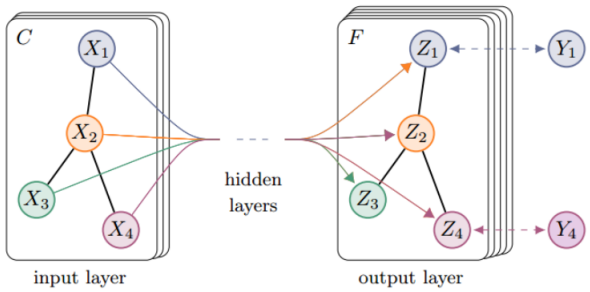

Figure 2: Graph Convolutional Neural Network

2 Related Work

Different kinds of deep neural networks have been taken advantage to extract the representation vector of the SMILES molecule expression:(1)Graph convolutional based method[2,4,5]. (2)CNN-based method. Smiles expressive signs could be embedded into vectors and compose the feature matrix, and CNN is used to extract features as fingerprints and they are fed into MLP to predict properties. (3)RNN-based method. Since smiles could be regarded as a sequence and RNN-based models such as seq2seq by Zheng Xu[3] and seq3seq by Xiaoyu Zhang[1] is proposed. (4)Attention-mechanism based method. Some work by Ashish Vaswani[6] takes the advantage of self-attention

mechanism in NLP. The latter three models could not very effectively extract the spatial feature of the molecule but the first nomal GCN is also trapped into low expressiveness of input feature matrix, therefore we propose a GCN+self-attention based model which could solve these two problems.

3 Methodology

Our model is composed of four kinds of layers:

- input layer
- graph embedding layer
- fully connected layer
- output layer.

3.1 Input layer with embedding

For graph-convolutional nerual network, except from the Laplacian matrix to represent the spatial relationship between one node and its neighbor, we should also define the input matrix where every sequential atom is represented by a feature vector and these vectors are concatenated to form a matrix. Inspired from the work by Maya Hirohara[7], we construct our feature matrix using RDkit in the following way.

Table 1: Input Features

Feature	Description	Size
Atom type	C,H,N,O or other types	53
Degree	the degree of unsaturation	7
NumHs	num of Hs in the neighborhood	5
Valence	its total valence	4
Aromaticity	wether the atom is aromatic	1
hybridization	ways of hybridization sp,sp2,sp3...	7
ring	wether the atom is in the ring	1

Naturally, we could use the common one-hot representations to represent every different atom/Degree/NumHs/Valence/hybriddization as following.

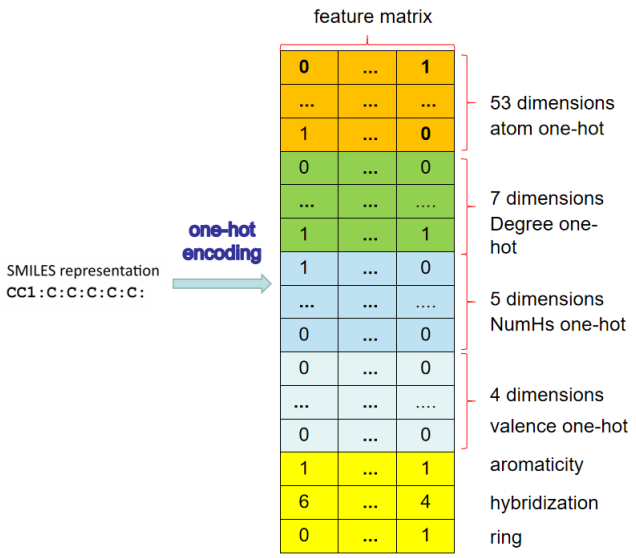

Figure 3: One-hot Encoding

However, there is a deadful drawback on this kind of representation:no similarity between different types can be learned because their one-hot representation are orthogonal in high-dimensional feature space. For example, halogen element Cl,Br,I enjoy a high resemblance in many properties but they could only be regarded as totally diferent.
Inspired by attention mechanism[6], in the input layer, we replace the one-hot encoding with a self-learning attention layer(simplified as embedidng layer), which is sealed as torch.nn.embedding() in pytorch-1.1.0.
Essentially, the embedding layer uses a key matrix to transform the input type information into a vector. Interestingly, this key matrix itself is learned during the training process. By learning the mapping relationship, we could mine more underlying information between different atoms and their properties. The goal embedding dimension should be at least the num of different types, otherwise two different atoms will enjoy a common representation vector, which drops out their difference.

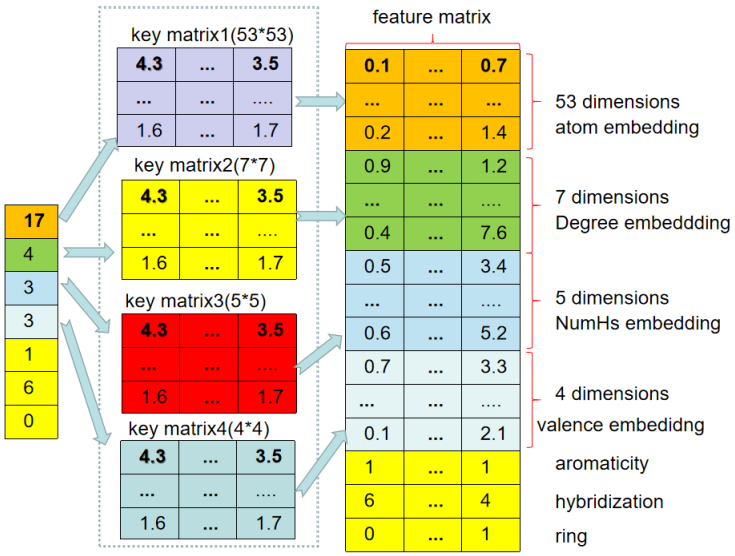

Figure 4: self-attention mechanism(embedding layer)

Therefore, we have now totally $53+7+5+4+1+1+1=72$ dimensional representation vector for every atom. Since the maximum length for a molecule in our dataset is 132, we should pad shorter atom sequence with 0(None) to length 132, and finally we could get the 132×72 input matrix.

3.2 Graph-embedding layer

To avoid overfitting, We construct a three-layer GCN for semi-supervised node classification on a graph with a symmetric adjacency matrix A to learn the internal relationship between every node and its neighborhood. Firstly we should get symetric adjacent matrix by adding the adjacent matrix and degree matrix.

$$\hat{A} = D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}$$

Then our forwarding process could be written as

$$Z^{(0)} = ReLU(\hat{A}XW^{(0)})$$

$$Z^{(1)} = ReLU(\hat{A}Z^{(0)}W^{(1)})$$

$$Z^{(2)} = ReLU(\hat{A}Z^{(1)}W^{(2)})$$

The output feature matrix is then fed into the fully connected layer.

3.3 Fully connected layer and output layer

Before we start to feed the feature into the linear layer, we shoudl first flatten it. However, taken every element of one atom column vector into consideration, it's better for us to first calculate average of every column, which is equivalent to the action that keep all weight parameters of all elements in one column remain the same (because they all belong to the same atom).

$$H = mean(Z^{(2)}, axis = 1)$$

Then it's time for us to feed it into linear layer, activation layer, dropout layer, another linear layer and finally pass the softmax layer to get the output.The whole scenario of the model is as follows.

3.4 Whole Scenario of Model

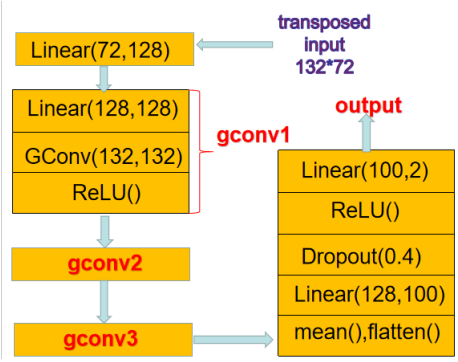

Figure 5: the whole network

4 Experimental Results

4.1 Implementation Details

The proposed GCN+self-attention mechanism(embedding) is implemented with the Pytorch, which is Facebook AI Research deep learning framework in Python. For more concenience and more effiency, we migrate our code to Huaweiyun Service Platform which is equipped with CUDA-10.1, anaconda3 and pytorch-1.1.0. As for the data preprocess of SMILES data, we use the RDkit toolkit API to extract the graphical structure from SMILES representation.

4.2 Experimental Settings

In the experiment, we use the Adam as the optimizer, binary cross entropy loss as the loss functioin and set the learning rate at 2.5e-4. For graph convolutional part, we could self-define the dimension of the Linear layer, here we take the num 128 as the weights matrix size in order to get a more parallel and fatser training. Otherwise, we set the training epoch as 50, the batch size as 32 to avoid overfitting and meanwhile we will calculate the validation loss to determine when to stop, as so-called early stopping technique.

4.3 Data Description

The dataset provided is composed of three file folders: train, validation and test, where training folder contains 8169 SMILES representations and its corresponding label, validation folder contains 272 SMILES representations and its corresponding label and validation folder contains 610 SMILES representations and no label. Since the ratio of training data to validation data is much less than 0.1, and their distribution is likely to be distinct from each other, so we decide to merge the training and validation data and randomly split the training and validation data with the split ratio 0.1. We find that the positive and negative sample in training and validation data bears a severe imbalance where the ratio could reach up to 9:1. To improve our precision and recall, we should generate more

positive samples and therefore, we take the data augmentation step by sampling positive samples and seal it with same amount of negative samples to reach a class balance in the dataloader. Finally,we get the training dataset of size 12883 and validation dataset of size 844. To speak of, some SMILES's molecular expression could not be extracted by RDkit, since totally there are only 5 samples or so ,we directly assign their input feature matrix with all 0.

4.4 Experimental Results

4.4.1 Training and Validation loss

For convergence performancce, we draw the training and validation loss value along 50 epoches. From the above figure, we could surprisingly find that the validation loss is always floating around

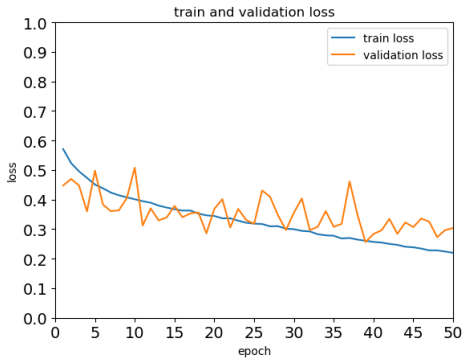

Figure 6: Training and validation loss comparison

the training loss, which demonstrates that during these 50 epoches, our model is hardly possible to overfit. It proves that our model has a very high robustness. And we could predict that the training and validation loss will continue to decrease as the epoches become higher.

4.4.2 Training and Validation Accuracy

For accuracy measurement, we draw the training and validation accuracy value along 50 epoches. This a really beatiful curve.From the above figure, we could find that the validation acuuracy is also

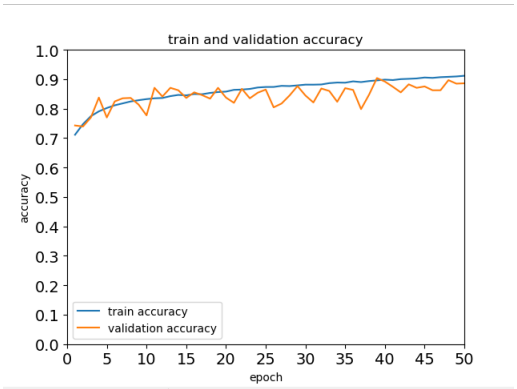

Figure 7: Training and validation accuracy comparison

getting really close to the training accuracy. The inexistence of the gap means that our training error is really close to the generalization error, which means we could predict that our test data auc on kaggle by calculating this train and validation auc result. It turns out to be at least 0.88 and often be over 0.90. After measuring epoch 20,30,40 and 50 epochs' training model on kaggle we could find that we reach the highest performance over most of other competitors and it still has great potential to rise up.

Table 2: Test result

epoch	Test auc
20	0.881
30	0.889
40	0.893
50	0.901

4.5 Comparison With Other Methods

Since we implement our self-attention mechanism to enhance the expressiveness of input feature matrix, we will directly compare it with the one-hot embedding input feature matrix.

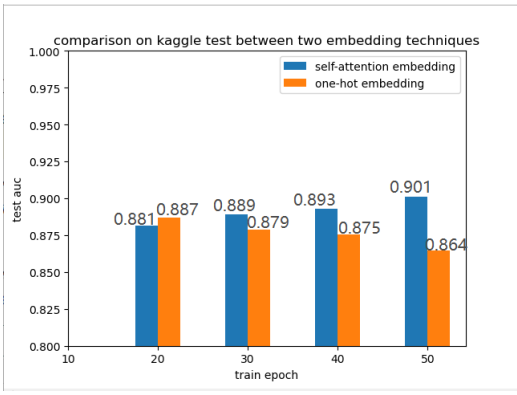

Figure 8: the whole network

From the above figure, we could find that although at first the one-hot embedding feature matrix enjoys a higher test auc than self-attention embedding layer, after training more and more epoches, the former one starts to lower down beacause of overfitting but the latter one start to steadily increase. This illustrates that self-attention mechanism has a stronger expressiveness than simple one-hot embedding, this is because one-hot vector rules the mapping relationship in advance and forbid the representation to learn the similarity between different atoms. On the contrary, the self-attention embedding layer will learn the key transformation matrix from the data and label and could better fit specific task. Therefore, as the training steps forward, the low-expressive one-hot model begin to overfit but high-expressive self-attention still goes up.

5 Conclusion and Future Work

In this paper, we propse a new GCN+self-attention based neural network. Instead of using one-hot vector as the input feature matrix, we train the key transformation matrix and use self-attention mechanism to generate input feature vector. The result on the kaggle competition test proves the superiority of our model than most of competitors. In the future, we will explore how to learn the weight of different connections. At present, we still stop at the stage when we assign every edge bond with 1,but obviously different bonds will have a different influence on its atoms.

6 Acknowledgement

This work is partially supported by the SJTU CS410 class Professor Shuai Li and TA Qizhi Li. They provided part of the SMILES-toxicity dataset. The author would like to thank NVIDIA for GPU donation and Huaweiyun cloud server platform.

References

[1]Xiaoyu Zhang, Sheng Wang, Feiyun Zhu, Zheng Xu, Yuhong Wang, and Jun zhou Huang. 2018. Seq3seq fingerprint: towards end-to-end semi-supervised deep drug discovery. In Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics. ACM, 404413.

[2]Dzmitry Bahdanau,Kyunghyun Cho,and Yoshua Bengio. 2014. Neural , machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014).

[3]Zheng Xu, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2017. Seq2seq Fingerprint: An Unsupervised Deep Molecular Embedding for Drug Discovery. InBCB

[4]Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016).

[5]Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive Graph Convolutional Neural Networks. arXiv preprint arXiv:1801.03226 (2018).

[6]Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems. 59986008.

[7]Maya Hirohara1, Yutaka Saito2,3, Yuki Koda1, Kengo Sato1 and Yasubumi Sakakibara1* Convolutional neural network based on SMILES representation of compounds for detecting chemical motif. From 29th International Conference on Genome Informatics Yunnan, China. 35 December 2018